Claims

What is claimed is:

1. A system for executing managed and unmanaged code, comprising:

a garbage collection service that facilitates memory management; and

an execution engine that declares at least one reference as pinned during execution of managed and unmanaged code, the execution engine notifying the garbage collection service of the at least one pinned reference, so that objects associated with the at least one pinned reference are not relocated during the garbage collection service.

2. The system of claim 1, the at least one reference being local variables referencing managed code during a call to unmanaged code.

3. The system of claim 1, further comprising a code manager operable to identify a set of roots of an executing program on a process stack, the set of roots comprising both references and pinned references.

4. The system of claim 3, the references being at least one of object references and interior references and the pinned references being at least one of pinned object references and pinned interior references.

5. The system of claim 4, the object references being pointers to object headers residing in the memory heap and the interior references being pointers to data members residing in object bodies in the memory heap.

6. The system of claim 3, the garbage collection service being operable to receive the references and pinned references from the code manager and trace through a memory heap to determine which memory segments are accessible and inaccessible, the garbage collection service then removing inaccessible objects, moving accessible objects that are not referenced by pinned references and holding objects fixed in memory that are

referenced by pinned references.

7. The system of claim 3, the code manager and garbage collection service being invoked by a request for memory by the executing code.

8. The system of claim 3, the code manager and garbage collection service being invoked periodically.

9. The system of claim 3, the code manager and garbage collection service being invoked when the memory heap becomes full.

10. The system of claim 3, the code manager and garbage collection service being part of a garbage collection system.

11. The system of claim 3, the garbage collection service being further operable to shift the non-garbage objects down in the memory heap to eliminate the inaccessible segments and to modify the program roots so that the references refer to the new locations of the objects in the memory heap, while holding fixed in memory non-garbage objects referenced by pinned references.

12. The system of claim 1, the execution engine operable to compile source code, the at least one pinned reference being declared as type pinned in the source code.

13. The system of claim 12, the execution engine comprising a just-in-time compiler adapted to compile both managed source code and unmanaged source code in real-time.

14. The system of claim 1, the pinned references being limited to residing on the stack.

15. A system for executing code in real-time, the system comprising:

a compiler operable to compile source code into an intermediate definition language and then into executable code for executing one or more programs, the compiler declaring at least one reference as pinned during compiling of the source code into the intermediate definition language;

a processor operable to execute the executable code and store objects within a memory heap and references in a stack during execution of the executable code; and

a memory management system being operable to locate the references on a stack and objects referenced by the references and reclaim storage being utilized by unused objects upon invocation of the memory management system, the objects referenced by pinned references being held fixed in memory during the reclaiming of storage.

16. The system of claim 14, the at least one reference being local variables referencing managed code utilized during an executing call of unmanaged code.

17. The system of claim 14, the compiler being operable to compile managed and unmanaged code into the intermediate definition language and compile the intermediate definition language into executable code.

18. The system of claim 14, the memory manager system comprising a code manager operable to locate the references and a garbage collector operable to reclaim storage being utilized by unused objects.

19. The system of claim 14, memory manager system being invoked by a request for memory by the executing code and the compiler allocating memory to the executing code after the memory manager has located the references and pinned references and reclaimed storage being utilized by unused objects.

20. The system of claim 14, the compiler being a just-in-time compiler adapted to compile both managed source code and unmanaged source code in real-time.

21. The system of claim 14, the pinned references being limited to residing on the stack.

22. A method of garbage collection during execution of code on a machine, comprising:

declaring at least one reference as pinned based on an unsafe condition;

receiving a request for a garbage collection service;

identifying references and pinned references in a process stack;

tracing a memory heap employing the references and pinned references to determine inaccessible memory objects;

removing the inaccessible memory objects from the heap; and

relocating objects in the heap except for objects referenced by pinned references.

23. The method of claim 21, the pinned references being local variables referencing managed code during an executing call of unmanaged code.

24. The method of claim 21, the execution of code comprising executing both managed and unmanaged code.

25. The method of claim 21, the references and pinned references including interior references and the memory heap being segmented into a plurality of blocks wherein the garbage collection service traces interior references by scanning through the blocks until the block with the object containing the interior reference is found.

26. The method of claim 24, further comprising adding objects found accessible during the tracing of the memory heap to a graph.

27. The method of claim 21, wherein removing the inaccessible memory objects from the heap comprises shifting non-garbage objects down in the memory heap to eliminate inaccessible segments and modifying the references to refer to the new locations of the objects in the memory heap, while holding fixed in memory non-garbage

objects referenced by pinned references.

28. A computer-readable medium having computer-executable instructions for performing the steps comprising:

executing code corresponding to at least one program;

storing objects on a memory heap during execution of the code;

storing references to the objects in a process stack during execution of the code;

declaring at least one reference as pinned and storing the at least one pinned reference on the process stack;

invoking a garbage collection algorithm;

scanning the stack to identify the object references and the at least one pinned reference in response to invocation of the garbage collection algorithm;

tracing the heap for inaccessible objects corresponding to the object references and the at least one pinned reference;

removing objects that have been determined to be inaccessible; and

relocating objects in the heap except for objects referenced by the at least one pinned reference.

29. The computer readable medium of claim 27, the code being both managed and unmanaged code.

30. The computer readable medium of claim 28, further comprising compiling source code by a just-in-time compiler in real-time prior to executing the code.

31. The computer readable medium of claim 29, wherein scanning the stack to identify the object references and at least one pinned reference is performed by a code manager.

32. The computer readable medium of claim 30, the code manger reporting the object references and at least one pinned reference to the garbage collection service, wherein the garbage collection service traces the heap for inaccessible objects

corresponding to the object references and at least one pinned reference and removes objects that have been determined to be inaccessible.

33. A system for garbage collecting during execution of code on a machine, comprising:

means for declaring references as pinned in response to an unsafe condition;

means for storing the pinned references on a process stack;

means for identifying object references and pinned references in the process stack; and

means for performing a garbage collection, such that objects referenced by pinned references remained in a fixed location in a heap.

34. The system of claim 32, further comprising means for compiling both managed and unmanaged code in real-time and means for storing object references in the process stack and means for storing objects in the memory heap.